# Package: weathercan (via r-universe)

November 13, 2024

**Type** Package

**Title** Download Weather Data from Environment and Climate Change Canada

**Version** 0.7.2

**Description** Provides means for downloading historical weather data from the Environment and Climate Change Canada website (<https://climate.weather.gc.ca/historical_data/search_historic_data_e.html>). Data can be downloaded from multiple stations and over large date ranges and automatically processed into a single dataset. Tools are also provided to identify stations either by name or proximity to a location.

**License** GPL-3

**Language** en-CA

**BugReports** https://github.com/ropensci/weathercan/issues/

**LazyData** TRUE

**URL** https://docs.ropensci.org/weathercan/, https://github.com/ropensci/weathercan/

**Depends** R (>= 4.1.0)

**Imports** dplyr (>= 1.0.0), httr (>= 1.4.2), lubridate (>= 1.7.1), memoise (>= 2.0.0), methods (>= 3.2.2), purrr (>= 0.3.4), rlang (>= 0.1.4), readr (>= 2.0.0), rvest (>= 0.3.4), stringi (>= 1.1.2), stringr (>= 1.4.0), tidyr (>= 1.1.3), tidyselect (>= 1.0.0), xml2 (>= 0.1.2), rappdirs (>= 0.3.3)

**RoxygenNote** 7.3.1

**Roxygen** list(markdown = TRUE)

**Suggests** devtools, ggplot2, htmltools, knitr, leaflet, lutz, mockery, naniar, rmarkdown, sf, testthat (>= 3.0.0), vcr (>= 1.0.2), withr

**VignetteBuilder** knitr

**Encoding** UTF-8

**Config/testthat/edition** 3

**Config/pak/sysreqs** libicu-dev libxml2-dev libssl-dev libx11-dev

**Repository** https://ropensci.r-universe.dev

**RemoteUrl** https://github.com/ropensci/weathercan

**RemoteRef** main

**RemoteSha** 1eaf3963e5cfec1890d1b4c68d09e0e196f804e9

# Contents



---

`check_eccc` *Check access to ECCC*

---

## Description

Checks if whether there is internet access, weather data, normals data, and eccc sites are available and accessible, and whether we're NOT running on cran

## Usage

```
check_eccc()
```

## Value

FALSE if not, TRUE if so

## Examples

```
check_eccc()
```

## codes *Meaning of climate normal 'codes'*

### Description

A reference dataset containing codes matched to their meaning. Data downloaded using the normals_dl() function contains columns indicating code. These are presented here for interpretation.

### Usage

```
codes
```

### Format

A data frame with 4 rows and 2 variables:

**code** Code

**meaning** Explanation of the code

## finches *RFID Data on finch visits to feeders*

### Description

RFID Data on finch visits to feeders

### Usage

```
finches
```

### Format

An example dataset of finch RFID data for interpolation:

**bird_id** Bird ID number

**time** Time

**feeder_id** feeder ID

**species** Species

**lat** Latitude of station location in degree decimal format

**lon** Longitude of station location in degree decimal format

## flags *Meaning of coded 'flags'*

### Description

A reference dataset containing 'flags' matched to their meaning. Data downloaded using the `weather_dl()` function contains columns indicating 'flags' these codes are presented here for interpretation.

### Usage

```
flags
```

### Format

A data frame with 16 rows and 2 variables:

**code** Flag code

**meaning** Explanation of the code

## glossary *Glossary of units and terms*

### Description

A reference dataset matching information on columns in data downloaded using the `weather_dl()` function. Indicates the units of the data, and contains a link to the ECCC glossary page explaining the measurement.

### Usage

```
glossary
```

### Format

A data frame with 77 rows and 5 variables:

**interval** Data interval type, 'hour', 'day', or 'month'.

**ECCC_name** Original column name when downloaded directly from ECCC

**weathercan_name** R-compatible name given when downloaded with the `weather_dl()` function using the default argument `format = TRUE`.

**units** Units of the measurement.

**ECCC_ref** Link to the glossary or reference page on the ECCC website.

## glossary_normals *Glossary of terms for Climate Normals*

### Description

A reference dataset matching information on columns in climate normals data downloaded using the `normals_dl()` function. Indicates the names and descriptions of different data measurements.

### Usage

```
glossary_normals
```

### Format

A data frame with 18 rows and 3 variables:

**ECCC_name** Original measurement type from ECCC

**weathercan_name** R-compatible name given when downloaded with the `normals_dl()` function

**description** Description of the measurement type from ECCC

## kamloops *Hourly weather data for Kamloops*

### Description

Downloaded with `weather()`. Terms are more thoroughly defined here https://climate.weather.gc.ca/glossary_e.html

### Usage

```
kamloops
```

### Format

An example dataset of hourly weather data for Kamloops:

**station_name** Station name

**station_id** Environment Canada's station ID number. Required for downloading station data.

**prov** Province

**lat** Latitude of station location in degree decimal format

**lon** Longitude of station location in degree decimal format

**date** Date

**time** Time

**year** Year

**month** Month

**day** Day

**hour** Hour

**qual** Data quality

**weather** The state of the atmosphere at a specific time.

**hmdx** Humidex

**hmdx_flag** Humidex data flag

**pressure** Pressure (kPa)

**pressure_flag** Pressure data flag

**rel_hum** Relative humidity

**rel_hum_flag** Relative humidity data flag

**temp** Temperature

**temp_dew** Dew Point Temperature

**temp_dew_flag** Dew Point Temperature flag

**visib** Visibility (km)

**visib_flag** Visibility data flag

**wind_chill** Wind Chill

**wind_chill_flag** Wind Chill flag

**wind_dir** Wind Direction (10's of degrees)

**wind_dir_flag** wind Direction Flag

**wind_spd** Wind speed km/hr

**wind_spd_flag** Wind speed flag

**elev** Elevation (m)

**climate_id** Climate identifier

**WMO_id** World Meteorological Organization Identifier

**TC_id** Transport Canada Identifier

## Source

https://climate.weather.gc.ca/index_e.html

`kamloops_day` *Daily weather data for Kamloops*

## Description

Downloaded with `weather()`. Terms are more thoroughly defined here https://climate.weather.gc.ca/glossary_e.html

## Usage

```
kamloops_day
```

## Format

An example dataset of daily weather data for Kamloops:

**station_name** Station name

**station_id** Environment Canada's station ID number. Required for downloading station data.

**prov** Province

**lat** Latitude of station location in degree decimal format

**lon** Longitude of station location in degree decimal format

**date** Date

**year** Year

**month** Month

**day** Day

**cool_deg_days** Cool degree days

**cool_deg_days_flag** Cool degree days flag

**dir_max_gust** Direction of max wind gust

**dir_max_gust_flag** Direction of max wind gust flag

**heat_deg_days** Heat degree days

**heat_deg_days_flag** Heat degree days flag

**max_temp** Maximum temperature

**max_temp_flag** Maximum temperature flag

**mean_temp** Mean temperature

**mean_temp_flag** Mean temperature flag

**min_temp** Minimum temperature

**min_temp_flag** Minimum temperature flag

**snow_grnd** Snow on the ground (cm)

**snow_grnd_flag** Snow on the ground flag

**spd_max_gust** Speed of the max gust km/h

**spd_max_gust_flag** Speed of the max gust flag

**total_precip** Total precipitation (any form)

**total_precip_flag** Total precipitation flag

**total_rain** Total rain (any form)

**total_rain_flag** Total rain flag

**total_snow** Total snow (any form)

**total_snow_flag** Total snow flag

**elev** Elevation (m)

**climate_id** Climate identifier

**WMO_id** World Meteorological Organization Identifier

**TC_id** Transport Canada Identifier

## Source

https://climate.weather.gc.ca/index_e.html

---

`normals_dl` *Download climate normals from Environment and Climate Change Canada*

---

## Description

Downloads climate normals from Environment and Climate Change Canada (ECCC) for one or more stations (defined by `climate_ids`). For details and units, see the `glossary_normals` data frame or the `glossary_normals` vignette: `vignette("glossary_normals", package = "weathercan")`

## Usage

```
normals_dl(
  climate_ids,
  normals_years = "1981-2010",
  format = TRUE,
  stn = NULL,
  verbose = FALSE,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| `climate_ids` | Character. A vector containing the Climate ID(s) of the station(s) you wish to download data from. See the `stations` data frame or the `stations_search` function to find Climate IDs. |
| `normals_years` | Character. The year range for which you want climate normals. Default "1981-2010". One of "1971-2000", "1981-2010", "1991-2020". Note: Some "1991-2020" are available online, but are not yet downloadable via weathercan. |

| | |
|---|---|
| format | Logical. If TRUE (default) formats measurements to numeric and date accordingly. Unlike weather_dl(), normals_dl() will always format column headings as normals data from ECCC cannot be directly made into a data frame without doing so. |
| stn | DEFUNCT. Now use stations_dl() to update internal data and stations_meta() to check the date it was last updated. |
| verbose | Logical. Include progress messages |
| quiet | Logical. Suppress all messages (including messages regarding missing data, etc.) |

## Details

Climate normals from ECCC include two types of data, averages by month for a variety of measurements as well as data relating to the frost-free period. Because these two data sources are quite different, we return them as nested data so the user can extract them as they wish. See examples for how to use the unnest() function from the tidyr package to extract the two different datasets.

The data also returns a column called meets_wmo this reflects whether or not the climate normals for this station met the WMO standards for temperature and precipitation (i.e. both have code >= A). Each measurement column has a corresponding _code column which reflects the data quality of that measurement (see the 1991-2020, 1981-2010, or 1971-2000 for more details) ECCC calculation documents.

Climate normals are downloaded from the url stored in option weathercan.urls.normals. To change this location use: options(weathercan.urls.normals = "your_new_url").

## Value

tibble with nested normals and first/last frost data

## Examples

```
# Find the climate_id
stations_search("Brandon A", normals_years = "current")

# Download climate normals 1981-2010
n <- normals_dl(climate_ids = "5010480")
n

# Pull out last frost data *with* station information
library(tidyr)
f <- unnest(n, frost)
f

# Pull out normals *with* station information
nm <- unnest(n, normals)
nm

# Download climate normals 1971-2000
n <- normals_dl(climate_ids = "5010480", normals_years = "1971-2000")
```

```
n

# Note that some do not have last frost dates
n$frost

# Download multiple stations for 1981-2010,
n <- normals_dl(climate_ids = c("301C3D4", "301FFNJ", "301N49A"))
unnest(n, frost)


# Note, putting both normals and frost data into the same data set can be done but makes for
# a very unweildly dataset (there is lots of repetition)
nm <- unnest(n, normals) |>
  unnest(frost)
```

---

`normals_measurements` *List of climate normals measurements for each station*

---

## Description

A data frame listing the climate normals measurements available for each station.

## Usage

```
normals_measurements
```

## Format

A data frame with 113,325 rows and 5 variables:

**prov** Province

**station_name** Station Name

**climate_id** Climate ID

**normals** Year range of climate normals

**measurement** Climate normals measurement available for this station

pg *Hourly weather data for Prince George*

## Description

Downloaded with `weather()`. Terms are more thoroughly defined here https://climate.weather.gc.ca/glossary_e.html

## Usage

```
pg
```

## Format

An example dataset of hourly weather data for Prince George:

**station_name** Station name

**station_id** Environment Canada's station ID number. Required for downloading station data.

**prov** Province

**lat** Latitude of station location in degree decimal format

**lon** Longitude of station location in degree decimal format

**date** Date

**time** Time

**year** Year

**month** Month

**day** Day

**hour** Hour

**qual** Data quality

**weather** The state of the atmosphere at a specific time.

**hmdx** Humidex

**hmdx_flag** Humidex data flag

**pressure** Pressure (kPa)

**pressure_flag** Pressure data flag

**rel_hum** Relative humidity

**rel_hum_flag** Relative humidity data flag

**temp** Temperature

**temp_dew** Dew Point Temperature

**temp_dew_flag** Dew Point Temperatureflag

**visib** Visibility (km)

**visib_flag** Visibility data flag

**wind_chill** Wind Chill

**wind_chill_flag** Wind Chill flag

**wind_dir** Wind Direction (10's of degrees)

**wind_dir_flag** wind Direction Flag

**wind_spd** Wind speed km/hr

**wind_spd_flag** Wind speed flag

**elev** Elevation (m)

**climate_id** Climate identifier

**WMO_id** World Meteorological Organization Identifier

**TC_id** Transport Canada Identifier

## Source

https://climate.weather.gc.ca/index_e.html

---

`stations` *Access Station data downloaded from Environment and Climate Change Canada*

---

## Description

This function access the built-in stations data frame. You can update this data frame with `stations_dl()` which will update the locally stored data.

## Usage

```
stations()
```

## Format

A data frame:

**prov** Province

**station_name** Station name

**station_id** Environment Canada's station ID number. Required for downloading station data.

**climate_id** Climate ID number

**WMO_id** Climate ID number

**TC_id** Climate ID number

**lat** Latitude of station location in degree decimal format

**lon** Longitude of station location in degree decimal format

**elev** Elevation of station location in metres

**tz** Local timezone excluding any Daylight Savings

**interval** Interval of the data measurements ('hour', 'day', 'month')

**start** Starting year of data record

**end** Ending year of data record

**normals** Whether *any* climate normals are available for that station (new behaivour)

**normals_1991_2020** Whether 1991-2020 climate normals are available for that station. **Note** that even if available, these are not yet downloadable via weathercan.

**normals_1981_2010** Whether 1981-2010 climate normals are available for that station

**normals_1971_2000** Whether 1971-2000 climate normals are available for that station

## Details

You can check when this was last updated with `stations_meta()`.

A dataset containing station information downloaded from Environment and Climate Change Canada. Note that a station may have several station IDs, depending on how the data collection has changed over the years. Station information can be updated by running `stations_dl()`.

## Source

https://climate.weather.gc.ca/index_e.html

## Examples

```
stations()
stations_meta()

# Which Manitoba stations have *any* climate normals?

library(dplyr)
filter(stations(), interval == "hour", normals == TRUE, prov == "MB")
```

---

# stations_dl *Get available stations*

---

## Description

This function can be used to download a Station Inventory CSV file from Environment and Climate Change Canada. This is only necessary if the station you're interested was only recently added. The 'stations' data set included in this package contains station data downloaded when the package was last compiled. This function may take a few minutes to run.

## Usage

```
stations_dl(skip = NULL, verbose = FALSE, quiet = FALSE)
```

### Arguments

| | |
|---|---|
| skip | Numeric. Number of lines to skip at the beginning of the csv. If NULL, automatically derived. |
| verbose | Logical. Include progress messages |
| quiet | Logical. Suppress all messages (including messages regarding missing data, etc.) |

### Details

The stations list is downloaded from the url stored in the option `weathercan.urls.stations`. To change this location use `options(weathercan.urls.stations = "your_new_url")`.

The list of which stations have climate normals is downloaded from the url stored in the option `weathercan.urls.stations.normals`. To change this location use `options(weathercan.urls.normals = "your_new_url")`.

Currently there are two sets of climate normals available: 1981-2010 and 1971-2000. Whether a station has climate normals for a given year range is specified in `normals_1981_2010` and `normals_1971_2000`, respectively.

The column `normals` represents the most current year range of climate normals (i.e. currently 1981-2010)

### Examples

```
# Update stations data frame
stations_dl()

# Updated stations data frame is now automatically used
stations_search("Winnipeg")
```

---

## stations_meta *Show stations list meta data*

---

### Description

Date of ECCC update and date downloaded via weathercan.

### Usage

```
stations_meta()
```

### Examples

```
stations_meta()
```

stations_search *Search for stations by name or location*

## Description

Returns stations that match the name provided OR which are within `dist` km of the location provided. This is designed to provide the user with information with which to decide which station to then get weather data from.

## Usage

```
stations_search(
  name = NULL,
  coords = NULL,
  dist = 10,
  interval = c("hour", "day", "month"),
  normals_years = NULL,
  normals_only = NULL,
  stn = NULL,
  starts_latest = NULL,
  ends_earliest = NULL,
  verbose = FALSE,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| `name` | Character. A vector of length 1 or more with text against which to match. Will match station names that contain all components of `name`, but they can be in different orders and separated by other text. |
| `coords` | Numeric. A vector of length 2 with latitude and longitude of a place to match against. Overrides `lat` and `lon` if also provided. |
| `dist` | Numeric. Match all stations within this many kilometres of the `coords`. |
| `interval` | Character. Return only stations with data at these intervals. Must be any of "hour", "day", "month". |
| `normals_years` | Character. One of `NULL` (default), `current`, `1991-2020`, `1981-2010`, or `1971-2000`. `current` returns only stations from the most recent *complete* normals year range (i.e. `1981-2010`). Default `NULL` does not filter by climate normals. Specific year ranges return stations with normals in that period. See Details for more specifics. |
| `normals_only` | DEPRECATED. Logical. Return only stations with climate normals? |
| `stn` | DEFUNCT. Now use `stations_dl()` to update internal data and `stations_meta()` to check the date it was last updated. |
| `starts_latest` | Numeric. Restrict results to stations with data collection beginning in or before the specified year. |

| | |
|---|---|
| ends_earliest | Numeric. Restrict results to stations with data collection ending in or after the specified year. |
| verbose | Logical. Include progress messages |
| quiet | Logical. Suppress all messages (including messages regarding missing data, etc.) |

## Details

To search by coordinates, users must make sure they have the sp package installed.

The current, most recent, climate normals year range is 1981-2010.

## Value

Returns a subset of the stations data frame which match the search parameters. If the search was by location, an extra column 'distance' shows the distance in kilometres from the location to the station. If no stations are found withing dist, the closest 10 stations are returned.

## Examples

```
stations_search(name = "Kamloops")
stations_search(name = "Kamloops", interval = "hour")

stations_search(name = "Ottawa", starts_latest = 1950, ends_earliest = 2010)

stations_search(name = "Ottawa", normals_years = "current")   # 1981-2010
stations_search(name = "Ottawa", normals_years = "1981-2010") # Same as above
stations_search(name = "Ottawa", normals_years = "1971-2000") # 1971-2010

if(requireNamespace("sf")) {
  stations_search(coords = c(53.915495, -122.739379))
}
```

---

# weather_dl *Download weather data from Environment and Climate Change Canada*

---

## Description

Downloads data from Environment and Climate Change Canada (ECCC) for one or more stations. For details and units, see the glossary vignette (vignette("glossary", package = "weathercan")) or the glossary online https://climate.weather.gc.ca/glossary_e.html.

## Usage

```
weather_dl(
  station_ids,
  start = NULL,
  end = NULL,
  interval = "hour",
  trim = TRUE,
  format = TRUE,
  string_as = NA,
  time_disp = "none",
  stn = NULL,
  encoding = "UTF-8",
  list_col = FALSE,
  verbose = FALSE,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| station_ids | Numeric/Character. A vector containing the ID(s) of the station(s) you wish to download data from. See the stations data frame or the stations_search function to find IDs. |
| start | Date/Character. The start date of the data in YYYY-MM-DD format (applies to all stations_ids). Defaults to start of range. |
| end | Date/Character. The end date of the data in YYYY-MM-DD format (applies to all station_ids). Defaults to end of range. |
| interval | Character. Interval of the data, one of "hour", "day", "month". |
| trim | Logical. Trim missing values from the start and end of the weather dataframe. Only applies if format = TRUE |
| format | Logical. If TRUE, formats data for immediate use. If FALSE, returns data exactly as downloaded from Environment and Climate Change Canada. Useful for dealing with changes by Environment Canada to the format of data downloads. |
| string_as | Character. What value to replace character strings in a numeric measurement with. See Details. |
| time_disp | Character. Either "none" (default) or "UTC". See details. |
| stn | DEFUNCT. Now use stations_dl() to update internal data and stations_meta() to check the date it was last updated. |
| encoding | Character. Text encoding for download. |
| list_col | Logical. Return data as nested data set? Defaults to FALSE. Only applies if format = TRUE |
| verbose | Logical. Include progress messages |
| quiet | Logical. Suppress all messages (including messages regarding missing data, etc.) |

## Details

Data can be returned 'raw' (format = FALSE) or can be formatted. Formatting transforms dates/times to date/time class, renames columns, and converts data to numeric where possible. If character strings are contained in traditionally numeric fields (e.g., weather speed may have values such as "< 30"), they can be replaced with a character specified by `string_as`. The default is NA. Formatting also replaces data associated with certain flags with NA (M = Missing).

Start and end date can be specified, but if not, it will default to the start and end date of the range (this could result in downloading a lot of data!).

For hourly data, timezones are always "UTC", but the actual times are either local time (default; `time_disp = "none"`), or UTC (`time_disp = "UTC"`). When `time_disp = "none"`, times reflect the local time without daylight savings. This means that relative measures of time, such as "night-time", "daytime", "dawn", and "dusk" are comparable among stations in different timezones. This is useful for comparing daily cycles. When `time_disp = "UTC"` the times are transformed into UTC timezone. Thus midnight in Kamloops would register as 08:00:00 (Pacific time is 8 hours behind UTC). This is useful for tracking weather events through time, but will result in odd 'daily' measures of weather (e.g., data collected in the afternoon on Sept 1 in Kamloops will be recorded as being collected on Sept 2 in UTC).

Files are downloaded from the url stored in `getOption("weathercan.urls.weather")`. To change this location use `options(weathercan.urls.weather = "your_new_url")`.

Data is downloaded from ECCC as a series of files which are then bound together. Each file corresponds to a different month, or year, depending on the interval. Metadata (station name, lat, lon, elevation, etc.) is extracted from the start of the most recent file (i.e. most recent dates) for a given station. Note that important data (i.e. station name, lat, lon) is unlikely to change between files (i.e. dates), but some data may or may not be available depending on the date of the file (e.g., station operator was added as of April 1st 2018, so will be in all data which includes dates on or after April 2018).

## Value

A tibble with station ID, name and weather data.

## Examples

```
kam <- weather_dl(station_ids = 51423,
                  start = "2016-01-01", end = "2016-02-15")

stations_search("Kamloops A$", interval = "hour")
stations_search("Prince George Airport", interval = "hour")

kam.pg <- weather_dl(station_ids = c(48248, 51423),
                     start = "2016-01-01", end = "2016-02-15")

library(ggplot2)

ggplot(data = kam.pg, aes(x = time, y = temp,
                          group = station_name,
                          colour = station_name)) +
```

```
  geom_line()
```

---

`weather_interp` *Interpolate and add weather data to a dataframe*

---

## Description

When data and the weather measurements do not perfectly line up, perform a linear interpolation between two weather measurements and merge the results into the provided dataset. Only applies to numerical weather columns (see `weather` for more details).

## Usage

```
weather_interp(
  data,
  weather,
  cols = "all",
  interval = "hour",
  na_gap = 2,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | Dataframe. Data with dates or times to which weather data should be added. |
| `weather` | Dataframe. Weather data downloaded with `weather` which should be interpolated and added to `data`. |
| `cols` | Character. Vector containing the weather columns to add or 'all' for all relevant columns. Note that some measure are omitted because they cannot be linearly interpolated (e.g., wind direction). |
| `interval` | What interval is the weather data recorded at? "hour" or "day". |
| `na_gap` | How many hours or days (depending on the interval) is it acceptable to skip over when interpolating over NAs (see details). |
| `quiet` | Logical. Suppress all messages (including messages regarding missing data, etc.) |

## Details

**Dealing with NA values** If there are NAs in the weather data, `na_gap` can be used to specify a tolerance. For example, a tolerance of 2 with an interval of "hour", means that a two hour gap in data can be interpolated over (i.e. if you have data for 9AM and 11AM, but not 10AM, the data between 9AM and 11AM will be interpolated. If, however, you have 9AM and 12PM, but not 10AM or 11AM, no interpolation will happen and data between 9AM and 12PM will be returned as NA.)

## Examples

```
# Weather data only
head(kamloops)

# Data about finch observations at RFID feeders in Kamloops, BC
head(finches)

# Match weather to finches
finch_weather <- weather_interp(data = finches, weather = kamloops)
```

# Index